MSc Assessed Prolog Lab Exercise

Issued: Wednesday 28 November 2012

Due: Wednesday 12 December 2012

1. Define a predicate *sentence/1*, and any other auxiliary predicates you need, such that *sentence(S)* succeeds if S is a correct grammatical sentence, according to the following grammar:

> *Sentence* consists of a *noun phrase* followed by *verb phrase*.
>
> *Noun phrase* is an *article* followed by a *noun*.
>
> *Verb phrase* is either a *verb*, or a *verb* followed by a *noun phrase*.

Your auxiliary predicates must include *noun_phrase/1* and *verb_phrase/1*.

Represent sentences as a list of words, e.g. [the, boy, eats, an, apple].

Test the program with your own lexicon that includes the following:

Nouns: grass, cow, girl, boy, apple, song,

Articles: the, a, an,

Verbs: eats, sings, chews, kicks.

Some example queries and answers:

?- sentence([a, cow, eats, the, grass]).	should get the answer yes.
?- sentence([the, girl, eats, an, apple]).	should get the answer yes.
?- sentence([the, boy, sings]).	should get the answer yes.
?- sentence([the, girl, sings, eats]).	should get the answer no.
?- sentence([girl, sings, a, song]).	should get the answer no.
?- sentence(S).	should get all instances of S that are grammatically correct. You will get many ridiculous sentences ([a grass eats a cow]), because the grammar is very simple – don't worry!

2. Modify your program for *noun_phrase/1* to a program *noun_phrase_better/1* so that it requires the article to match the noun in noun phrases, i.e. if the noun starts with a vowel then the article is either 'the' or 'an', not 'a'. So, for example, [a, apple] is not be recognized as a noun phrase.

Hint. Prolog has a built-in predicate *atom_chars/2* such that *atom_chars(W, L)* succeeds when *L* is the list of characters in the constant *W* in the order in which they appear in *W*. So *atom_chars(apple, L)* succeeds with *L=[a,p,p,l,e]*.

3. Extend the grammar so that your program can also deal with verb phrases that consist of a conjunction of adverbs, followed by a verb, followed by a noun phrase. An example of a sentence conforming to this extended grammar is " a cow slowly and deliberately chews the grass". When there are more than two adverbs, all but the final two are separated by commas, and only the final two are separated by the word "and". So, for example "slowly, deliberately and merrily" is recognized as a conjunction of adverbs, but not "slowly and deliberately and merrily".

Also make sure that your program does not allow repetitions of adverbs in the same phrase. So, for example a sequence of words such as "a cow slowly and slowly chews the grass" is not accepted or generated as a sentence. Use adverbs including the following: slowly, deliberately, merrily, sweetly.

4. Assume that Text is a conjunction of sentences according to the grammar of question 1 above, for example

[the, boy, chews, an apple, and, the girl, kicks, the boy].

Write a program for *actions*(*Actor*, *Text*, *As)* such that *As* is the list of actions that the *Actor* does according to the *Text*.

Test your program with the following queries:

actions(boy, [the, boy, chews, an apple, and, the girl, kicks, the, boy], X)

> should produce the answer X=[chews].

actions(apple, [the, boy, chews, an apple, and, the girl, kicks, the, boy], X)

> should produce the answer X=[].

actions(girl, [the, girl, chews, an apple, and, the girl, sings, a, song, and, the, girl, kicks, the, cow], X)

> should produce the answer X=[chews, sings, kicks]. The order of the elements in the list is not important.

5. Modify your program for *actions*(*Actor, Text, As)* to define a new predicate

actions_and_ adverbs(*Actor, Text, As)* such that *As* is the list of *(A, Advs)* where *A* is an action the *Actor* does and *Advs* is the list of adverbs associated with *A* according to the *Text*.

For example, given an English text:

> the boy slowly, merrily and sweetly eats an apple and the boy slowly kicks the girl and the boy merrily sings

the list of action and adverb pairs *(A, Advs)* for "boy" will be

> [(eats, [slowly,merrily,sweetly]), (kicks, [slowly]), (sings, [merrily])].

The order of the elements in the list is not important.

Each of the questions 1- 5 has 20% of the marks.

If your program requires the lists library add the following entry (called a Prolog *directive*) at the top of your file:

:- use_module(library(lists)).

This causes Sicstus to automatically load the lists library upon consulting your file.

Assessment Notes:

- Programs should work.
- Programs should not loop, but apart from that efficiency is not important.
- Clarity – use comments to explain your predicates.
- Good Prolog style – clear short rules.
- You probably do not need cuts. If you do use cuts you should make sure legitimate answers are not removed.
- Please make sure your programs include the predicates and the lexicon mentioned above, as these are important for auto-testing.
- Your programs should work if we change the lexicon.